1.7

IET Image Processing

Research Article

# Single image rain removal with reusing original input squeeze-and-excitation network

ISSN 1751-9659
Received on 27th June 2019
Revised 27th June 2019
Accepted on 15th January 2020
E-First on 13th May 2020
doi: 10.1049/iet-ipr.2019.0716
www.ietdl.org

*Meihua Wang*[1]*, Lunbao Chen*[1]*, Yun Liang*[1] ✉*, Yuexing Hao*[2]*, Haijun He*[1]*, Chao Li*[1]
[1]*College of Mathematics and Informatics, South China Agricultural University, Guangzhou, People's Republic of China*
[2]*Rutgers University–New Brunswick, New Jersey, USA*
✉ *E-mail: sdliangyun@163.com*

**Abstract:** In this study, the authors propose a novel network architecture to address the problem of removing rain streaks from single images. To strengthen the representational power of the network, they adopt the squeeze-and-excitation block in the network. Furthermore, they propose a new network connection called reusing original input (ROI). The ROI connection reuses the original input of the network and can provide more texture details of the background. These details can be useful for the restoration of the image after removing the rain streaks. Batch normalisation is applied to further improve the rain removal performance of the network. Despite the fact that the network is trained on synthetic data, experimental results show that the proposed network has a comparable performance on both synthetic images and real-world images to the state-of-the-art methods.

## 1 Introduction

Rain removal can be considered an important task in image denoising, image decomposition, and image restoration. It removes rain from images or videos to improve visual quality. With an input that covers rain, many computer vision algorithms, such as object tracking [1], might miss out on some significant features of the input, which will lead to the failure of the algorithms. For the outdoor vision system, the captured photographs or videos could suffer from several types of visibility degradation under rainy conditions. Raindrops blur and/or deform images like a magnifier. Distant rain streaks accumulate and result in foggy effects. Nearby rain streaks introduce image noise such as bright white streaks that blocks part of the scenes [2].

Lots of methods have been proposed for rain removal. These methods can be divided into two categories: video-based and single-image-based rain removal methods. Generally, removing rain from a single image is more challenging due to the lack of temporal information. However, it is important since in some cases, such as images downloaded from the internet, only one single rainy image is available. On the other hand, the video rain removal problem can be solved by applying the single-image-based method frame by frame.

Recently, deep learning methods have been widely used in various computer vision tasks such as image recognition [3–7], image de-noising [8–11], image classification [12], image de-hazing [13–16], and image super-resolution [17–19]. The effectiveness of the deep learning method has been well proved. As stated in [4], deeper networks would encounter the degradation problem without a proper network architecture. When used in image de-noising and de-hazing, deep learning methods might over smooth the images if the images contain textures similar to the noise or haze, i.e. they would remove some details of the images accidentally and thus blur the images.

In this work, a novel network architecture designed to tackle the single image rain removal task is proposed. The contributions of this work are three-fold.

(i) We propose a new network connection that can easily reuse the original input of the network. This connection is called reusing original input (ROI), and it is inspired by the residual learning network (ResNet) [4] and the densely connected convolutional network (DenseNet) [20]. ROI connection creates bypasses between the original input and the subsequent layers. These bypasses can be crucial for avoiding the degradation problem when the network goes deeper as stated in [4]. Furthermore, the ROI connection can provide more texture details for the network. These details are important for the restoration after rain streaks removal and can help to decrease the over-smoothness.
(ii) Based on the ROI connection, we adopt squeeze-and-excitation (SE) block and batch normalisation (BN) to construct the proposed ROISE network (ROISEN). With the SE block, our ROISEN can utilise the relationships between channels, which could strengthen the representational power of the network. Experimental results confirm the effectiveness of ROISEN.
(iii) We train the proposed ROISEN with two different kinds of inputs. The first one is training with the rainy images directly, and the other one is training with the detail layers of the rainy images, as used by other works [21–23]. We make a comparison between these two training methods. To our knowledge, this is the first deep learning-based rain removal work to conduct an explicit comparison between these two kinds of inputs.

The remainder of this paper is as follows: in Section 2, the related rain removal works are reviewed. We break them into two parts: video-based and single-image-based methods. In Section 3, we discuss the details of the proposed ROISEN, including the network architecture and implementation details. In Section 4, we show our experimental results and analyse them. Finally, the paper is summarised in Section 5.

## 2 Related work

As mentioned above, rain removal methods can be divided into two categories: video-based and single-image-based methods.

### 2.1 Video-based rain removal methods

Video-based rain removal methods take advantage of the temporal contents of the video. In 2004, Garg and Nayar developed a model for the visual appearance of rain [24]. Unfortunately, when the changes in intensities caused by the rain streaks are small, detecting and removing rain streaks using this method can be difficult. Bossu *et al.* proposed a method that first detected moving objects by background subtraction. Then, they used selection rules based on photometric and size to segment the potential rain pixels.

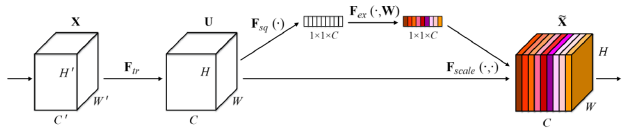

**Fig. 1** *SE block*

Next, a histogram of the orientation of streaks is built and then used to detect the rain pixels [25]. The frame rate of this method barely meets the real-time processing requirement. A similar problem exists in [26, 27]. In [26], Santhaseelan and Asari detected rain streaks based on phase congruency features and then restored the image utilising the intensities of the rainy pixels, spatial neighbours, and temporal neighbours. In [27], You *et al.* modelled the adherent raindrops and used the model for detection under the assumption that the raindrops are static during the detection process. Then, they removed the detected raindrops and restored the image differently for partially occluding areas and completely occluding areas. According to the study, only intensity-change based features can work in real time while the best performance of this method requires both intensity-change and motion-based features. In [28], Ren *et al.* divided rain streaks into sparse ones and dense ones and modelled them in a matrix decomposition framework. Then, they formulated the detection of sparse rain streaks and moving objects as a multi-label Markov random field while assuming that the dense rain streaks obey a Gaussian distribution. Finally, all rain streaks were removed using a low-rank representation of the backgrounds and rain pixels within the moving objects were filtered by a group sparsity term. This method may fail to separate dynamic textures and moving objects. A Computer Vision and Pattern Recognition (CVPR) work [29] proposed a hybrid rain model for both rain streaks and rain occlusion regions, and then they built a joint recurrent rain removal and reconstruction network to address the problem of video rain removal.

### *2.2 Single-image-based rain removal methods*

Owing to the lack of temporal information, video-based methods cannot be extended to single image rain removal directly. Also, it leads to the appearance of the single-image-based rain removal methods.

In 2014, Kim *et al.* proposed a method that first detected the rain streaks based on the assumption that the rain streaks have elongated elliptical shapes, then removed them using non-local mean filters [30]. This method works well in some situations. However, when the image contains rain streaks of different orientations or scales, the detection of rain streak could be challenging. Luo *et al.* used discriminative sparse coding to remove rain streaks in a single image [31]. This method performs better than Kim's work qualitatively but still not effective enough. In [2], Li *et al.* proposed a method that used patch-based priors, which are based on Gaussian mixture models (GMMs) for both the clean layers and rain layers. Both Luo's and Li's works tend to leave the marks of rain streaks after rain removal. Also, since the method proposed by Li has to learn the GMM first for the input rainy image, the running time could be longer depending on the size of the input image. Considering the intrinsic directional and structural information of rain streaks, Deng *et al.* proposed a global sparse model in [32] and solved it with an alternating direction method of multipliers.

In the past few years, some deep learning-based methods for single image rain removal are proposed. In 2017, Fu *et al.* proposed a network called DerainNet [21]. They first decomposed images into base layers and detail layers, then trained the DerainNet with detail layers. After rain removal, they utilised the image enhancement technique on both base layers and detail layers to further improve the rain removal results. This method works well when the rainy image has a foggy look. In the same year, Fu *et al.* proposed another network called DetailNet in [22]. Similarly, they used the same image decomposition technique used in [21] to prepare the training data. Also, this time, they proposed a negative residual mapping to ease the training process. The DetailNet performs well but it contains more layers than DerainNet, thus the training would be time-consuming. Xia *et al.* proposed a simplified residual dense network in [23] to address the problem. Also, they modified the image decomposition technique used in [21, 22] slightly, to help the network learn more accurately. Zhang and Patel proposed a density-aware multi-stream dense network, which they called DID-MDN in [33]. This network first learned the density information of rain, then removed rain streaks with the aid of this information. A multi-stream dense network was used to deal with rain streaks with different scales and shapes. According to the published codes, this method can only produce an output with a size of $512 \times 512$. In [34], the authors used a coarse network to generate a raw rain removal result and then used a refinement network to generate a better result. This method tends to blur the edges of some objects. All these methods have their own limitation, and there is still room for improvement.

## 3 Proposed method

In this section, we will take a closer look at the network architecture of the proposed ROISEN. First, we will briefly review the SE block, then we will introduce the ROI connection. After that, we will break down the structure of ROISEN for illustration. The implementation details will be discussed as well.

### *3.1 SE block*

SE block is proposed in [35] by Hu *et al*. According to the paper, the SE block aims at taking good advantage of the channel-wise information by explicitly modelling the interdependencies between the channels of feature maps. Fig. 1 is an illustration of the SE block. A SE block contains two major steps, squeeze and excitation, and they are denoted as $F_{\rm sq}(\cdot)$ and $F_{\rm ex}(\cdot, W)$, respectively, in Fig. 1. As can be seen in Fig. 1, there is an operation denoted as $F_{\rm tr}$. It represents any given transformation such as a convolution or a set of convolution. The output of this transformation is the input of the SE block. After the excitation, the SE block re-scales its input with the excitation output to form its final output. The re-scaling operation is denoted as $F_{\rm scale}(\cdot, \cdot)$. For any given transformation output $U$, $U \in \mathbb{R}^{H \times W \times C}$, a SE block can be described as follows:

$$z_c = F_{\rm sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j) \tag{1}$$

$$s = F_{\rm ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)) \tag{2}$$

$$\tilde{x}_c = F_{\rm scale}(u_c, s_c) = s_c \cdot u_c \tag{3}$$

In (1), $z_c$ is the descriptor of $u_c$, i.e. the $c$th channel of $U$. The SE block uses global average pooling to shrink $U$ through spatial dimensions $H \times W$, i.e. it uses the average value of each channel to represent the channel itself, and the average value is called the channel descriptor. In (2), $\sigma$ and $\delta$ represent the sigmoid activation and rectified linear unit (ReLU) function, respectively, $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, and $W_2 \in \mathbb{R}^{C \times C/r}$. $r$ is called the reduction ratio, which aims at limiting model complexity. Equation (3) describes the re-scaling operation of the SE block, where $F_{\rm scale}(u_c, s_c)$ refers to channel-wise multiplication. The final output of an SE block can be written as $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_C]$. The SE block can be considered as feature recalibration.

### *3.2 ROI connection*

Deeper networks would encounter the degradation problem without proper designs. According to [4], creating bypasses between layers can ease this problem. Another problem with deep learning methods is that they might accidentally remove some textures similar to the noise and thus blur the images, lead to over-smoothness. This might indicate that the network needs more features to distinguish the details of the images from the noise.

Instead of learning more features, which will introduce more parameters to the network, we can reuse the features extracted by the former layers as suggested in [20]. Inspired by ResNet [4] and DenseNet [20], we propose a new network connection that can easily reuse the original input of the network. We refer to this connection as an ROI connection. ROI connection creates bypasses between the original input and subsequent layers. This indicates that the latter layers of the network possess not only the features extracted from the input by the former layers but also the original input itself. Hence, an ROI connection can ease the degradation problem. Also, with the extra texture details, the network can learn more accurately and mitigate over-smoothness.

To implement an ROI connection, we concatenate the original input of the network and feature maps generated by the previous layer before entering the subsequent layer. To better understand the proposed ROI connection, we make a comparison among the residual connection in ResNet, the dense connection in DenseNet and our ROI connection. Fig. 2 shows these three connections. In the figure, '⊕' refers to element-wise addition and '©' refers to channel-wise concatenation. Note that the SE operation is not necessary for ROI connection. The differences between ROI connection and the other two connections can be summarised as follows:

(1) As shown in Fig. 2*a*, residual connection merges its inputs together via element-wise addition. This requires the inputs to have the same dimensions, including height, width, and depth. On the other hand, as shown in Fig. 2*c*, ROI connection utilises channel-wise concatenation for combination. One advantage of concatenation is that it does not require the inputs to have the same depth as long as they have the same height and width.

(2) As shown in Fig. 2*b*, for every concatenation, dense connection takes not only 'X', but also the output of the previous layer, and the outputs of the layers before as its inputs, while ROI connection takes only 'X' and the output of the previous layer for concatenation.

(3) In Figs. 2*a* and *b*, 'X' can be either the original input of the network or the output of the former layers. However, 'X' can only be the original input in Fig. 2*c* as an ROI connection only reuses the original input of the network.

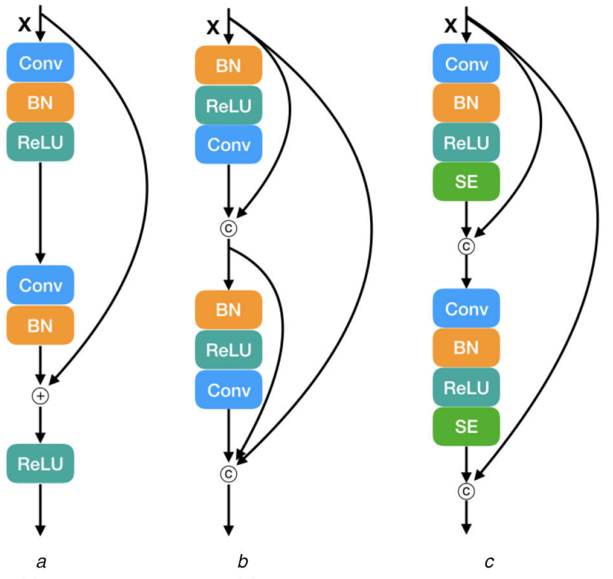

**Fig. 2** *Comparison of different network connections*
***(a)*** Residual, ***(b)*** Dense, ***(c)*** Proposed ROI

### 3.3 Proposed ROISEN

Fig. 3*b* shows the architecture of the proposed ROISEN. Note that the network contains building blocks called 'CBRS block', which is shown in Fig. 3*a*. A CBRS block contains a series of operations including **C**onvolution, **B**N, **R**eLU, and **S**E. We colour different operations with different colours for better distinction. Given an input *X*, the output of a CBRS block, *O*, can be obtained via the following equation:

$$O = \mathrm{CBRS}(X) = \mathrm{SE}(\max(0, \mathrm{BN}_{\gamma,\beta}(WX + b))) \tag{4}$$

In (4), $(WX + b)$ refers to convolution, with $W$ as weights and $b$ as biases. $\mathrm{BN}_{\gamma,\beta}(\cdot)$ represents BN with $\gamma$ and $\beta$ as scale and shift, respectively. ReLU is denoted as $\max(0, \cdot)$, and $\mathrm{SE}(\cdot)$ is the SE block.

As can be seen in Fig. 3*b*, the proposed ROISEN contains four CBRS blocks and one regular convolutional layer. Also, the red lines indicate the proposed ROI connections. Moreover, ROISEN does not have any pooling layers besides the global average pooling used in SE blocks. Since the pooling layer tends to reduce the size of the image too quickly, which is a severe problem for us since our training set contains images of small size. Furthermore, the pooling layer leaves out too much information, which we believe is against our goal as we want to utilise as much information as possible for image restoration after rain removal.

Given the network input *X*, the proposed ROISEN can be described as follows:

$$O_1 = \mathrm{CBRS}_1(X) \tag{5}$$

$$O_i = \mathrm{CBRS}_i([X, O_{i-1}]) \tag{6}$$

$$O_5 = \mathrm{conv}([X, O_4]) = W_5[X, O_4] + b_5 \tag{7}$$

where $i = 2, 3, 4$, and $[\cdot, \cdot]$ refers to channel-wise concatenation, i.e. the ROI connection in this work. $O_5$ is the final output of ROISEN.

### 3.4 Network implementation

In this part, we will discuss the implementation details of the proposed network.

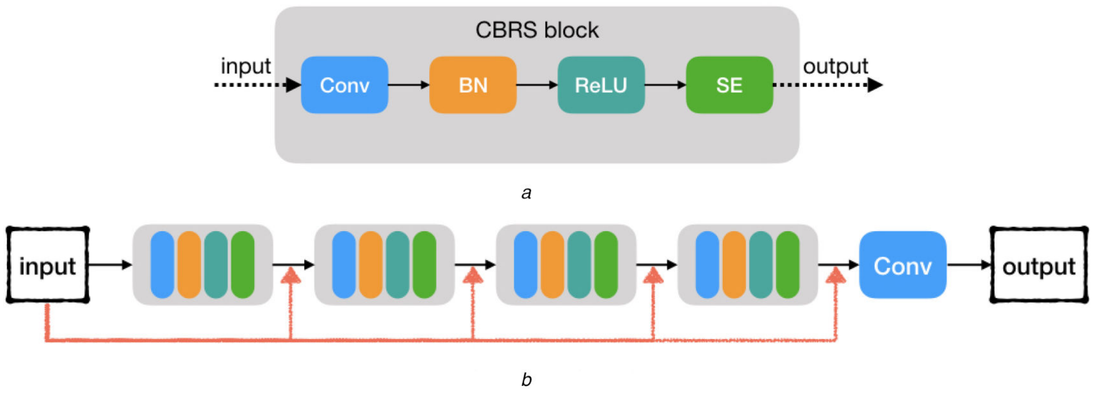

**Fig. 3** *Network architecture of ROISEN*
***(a)*** CBRS block, ***(b)*** Proposed ROISEN

*3.4.1 Data sets:* In order to train the proposed network, clear images and their corresponding rainy versions are needed. It is a big challenge to capture a rainy scene and its corresponding clear version in the real world due to the changing of the air light and object movements such as the leaves swayed by the wind. Therefore, a widely used trick is to first collect a large number of clear images of real-world scenes, and then synthesise their rainy version with rain streaks of different orientations following the instruction of the document on http://www.photoshopessentials.com/photo-effects/rain/.

With the trick introduced above, 127 clear images are collected and their rainy versions are synthesised for this work. Also, they are cut into a fixed size, which is $33 \times 33$ in this work. After the cut, we have around 96,200 image patches for training and 14,500 for validation.

We train the proposed ROISEN with different inputs, one is the original images and the other is the detail layers of images. To obtain the detail layers of all the images, we adopt the technique introduced in [23], i.e. instead of decomposing the rainy image and its corresponding clear image with two filters as in [21, 22], we only need one filter to get the detail layers of an image pair. More specifically, in [21, 22], for a rainy image $X$ and its corresponding clear image $Y$, the detail layer can be obtained by subtracting the base layer from the image, with the base layer obtained by a low-pass filter [36]. Then, the detail layers are used for training, while the base layers $X_{\text{base}}$ and $Y_{\text{base}}$ are omitted with the assumption that they are equal. This process can be described as follows:

$$X_{\text{detail}} = X - X_{\text{base}}, Y_{\text{detail}} = Y - Y_{\text{base}} \tag{8}$$

However, as stated in [23], Xia *et al.* found that the base layers are not equal. In fact, with the effect of rain, the pixel values in the base layer of the rainy image are greater than that of the clear image, i.e. $X_{\text{base}} > Y_{\text{base}}$. Therefore, when the network is trained to learn $X_{\text{detail}} = Y_{\text{detail}}$, the rain removal result will not be accurate since $(X_{\text{detail}} + X_{\text{base}}) > (Y_{\text{detail}} + Y_{\text{base}})$. To overcome this issue, Xia *et al.* used $X_{\text{base}}$ to replace $Y_{\text{base}}$, then the detail layer of the clear image can be obtained via the following equation:

$$Y'_{\text{detail}} = Y - X_{\text{base}} \tag{9}$$

Then, the rain removal result will be more accurate when the network is trained to learn $X_{\text{detail}} = Y'_{\text{detail}}$, since $(X_{\text{detail}} + X_{\text{base}}) = (Y'_{\text{detail}} + X_{\text{base}})$. According to the paper, this little change can accelerate training, and help the network learn the correct luminance information.

*3.4.2 Settings:* The parameters settings of convolutional layers in ROISEN are listed in Table 1. The setting of filter size is inspired by Dong *et al.* [17]. As can be seen in the table, we do not use ReLU in the last layer. It is an empirical setting that the activation function is not used in the last layer of a neural network. We utilise padding for all convolutional layers along with a unitary stride to ensure all intermediate feature maps and the final output have the same size as the input. To maintain the output size, for unitary stride, the padding size $P$ can be calculated via the following equation based on the filter size, $F \times F$:

$$P = \left\lceil \frac{F-1}{2} \right\rceil \tag{10}$$

*3.4.3 Training:* The training of the network aims at minimising the following loss function:

$$L(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \| F(X_i, \Theta) - Y_i \|^2 \tag{11}$$

In (11), we use $\Theta$ to represent all the parameters of the network. $F(\cdot, \Theta)$ represents the rain removal result, i.e. the output of the network. $X_i$ and $Y_i$ are the $i$th input rainy patch in the training set and the corresponding clear patch, respectively. $N$ denotes the total number of patches in the training set, which is 96,197 in this work.

The training process is carried out on a PC with deep learning framework Caffe [37], six CPUs of Intel i7-8700, 32 GB memory and a GPU of NVIDIA GeForce 1080.

## 4 Experiments

In this section, we will make a comparison between the proposed ROISEN and some other network architectures first. The difference in training the proposed network with two different kinds of inputs will be discussed as well. Then, we will carry out the comparison among different methods.

### *4.1 Different network architectures*

To demonstrate the effectiveness of the proposed ROISEN, we conduct an experiment between ROISEN and some other different network architectures. These network architectures are listed in Table 2. In the table, PlainCNN denotes the network that contains only convolutional layers and activation function, ROICNN refers to the network that adds ROI connections, ROISEN-NoBN represents the network that utilises ROI connections along with SE blocks but without BN, and ROISEN is the proposed network. We aim at revealing the improvements that ROI connection, SE block, and BN bring to the convolutional neural network, hence the settings of convolutional layers remain the same as listed in Table 1.

To quantitatively evaluate these networks, we run them on *Rain12* data sets [2] to get the peak signal-to-noise ratio (PSNR) [38] and structural similarity (SSIM) [39] values of their rain removal results. PSNR and SSIM are two widely used full-reference image quality assessments, and higher value indicates better quality. The average results are listed in Table 3. Note that all networks are trained with the rainy images directly, not the detail layers.

As can be learned from Table 3, PlainCNN has the lowest PSNR and SSIM values, while ROICNN scores are little higher. This indicates that the network can benefit from ROI connection. Compared to ROICNN, the PSNR and SSIM values of ROISEN-NoBN is higher, which shows the effectiveness of the SE block. Among these networks, the proposed ROISEN possesses the highest PSNR and SSIM values, which implies that ROISEN has the best rain removal result, proving that BN can be useful for rain removal from a single image. Though the increment of the SSIM

**Table 1** Parameters settings of convolutional layers

| Layers | Filter size | Filter number | Use ReLU | Padding |
|---|---|---|---|---|
| layer 1 | $9 \times 9$ | 128 | yes | 4 |
| layer 2 | $1 \times 1$ | 128 | yes | 0 |
| layer 3 | $5 \times 5$ | 64 | yes | 2 |
| layer 4 | $3 \times 3$ | 64 | yes | 1 |
| layer 5 | $3 \times 3$ | 3 | no | 1 |

**Table 2** Architectures of different networks

| Networks | ROI Connection | SE Block | BN |
|---|---|---|---|
| PlainCNN | no | no | no |
| ROICNN | yes | no | no |
| ROISEN-NoBN | yes | yes | no |
| ROISEN | yes | yes | yes |

**Table 3** Quantitative results of different networks on *Rain12* datasets

| Metrics | PlainCNN | ROICNN | ROISEN-NoBN | ROISEN |
|---|---|---|---|---|
| PSNR | 31.53 | 32.10 | 33.05 | **33.90** |
| SSIM | 0.9536 | 0.9549 | 0.9610 | **0.9616** |

value of ROISEN is minute compared to ROISEN-NoBN, the PSNR value has a notable improvement.

### 4.2 Results on different training inputs

We make an explicit comparison between two different training inputs. One is the rainy images, and the other one is the detail layers of rainy images. For unambiguous depiction, we refer to the network trained with rainy images directly as ROISEN-R, and the network trained with detail layers as ROISEN-D. We run these two networks on two data sets to get their PSNR and SSIM values. The first data set is the *Rain12* data sets, and the other one contains ten images that are randomly selected from the testing set used in [40, 41]. We refer to the latter data sets as *New10* data sets. The average results are listed in Table 4. Columns ‘PSNR1’ and ‘SSIM1’ list the PSNR and SSIM values of *Rain12* data sets, while ‘PSNR2’ and ‘SSIM2’ list that of, extitNew10 data sets.

As listed in the table, ROISEN-D scores the higher PSNR and SSIM values on both data sets compared to ROISEN-R, which means that the rain removal results of ROISEN-D have better quality than that of ROISEN-D. It also indicates that the proposed ROISEN would achieve better rain removal performance when trained with detail layers. One possible explanation is that when trained with detail layers, the network can focus better on the features of rain streaks as the detail layers contain only the texture information, leaving other information in the base layers. Based on this finding, we will compare ROISEN-D with state-of-the-art methods. Also, for consistency, we will use ROISEN to represent ROISEN-D, i.e. ROISEN refers to the proposed network trained with detail layers in the rest of this paper.

**Table 4** Quantitative results of different training inputs

| Networks | PSNR1 | SSIM1 | PSNR2 | SSIM2 |
|---|---|---|---|---|
| ROISEN-R | 33.90 | 0.9616 | 27.70 | 0.9126 |
| ROISEN-D | **34.80** | **0.9688** | **29.84** | **0.9310** |

**Table 5** Quantitative results of different methods

| Methods | PSNR1 | SSIM1 | PSNR2 | SSIM2 |
|---|---|---|---|---|
| DSC | 28.95 | 0.8836 | 26.25 | 0.8789 |
| LP | 32.21 | 0.9442 | 28.51 | 0.9128 |
| DetailNet | 31.45 | 0.9291 | 28.89 | 0.9309 |
| DID-MDN | 28.51 | 0.8901 | 24.37 | 0.8510 |
| DRCNN | 31.36 | 0.9485 | 27.82 | 0.9168 |
| ROISEN | **34.80** | **0.9688** | **29.84** | **0.9310** |

### 4.3 Results on synthetic images

To demonstrate the effectiveness of the proposed ROISEN, we compare our method with some state-of-the-art methods, including discriminative sparse coding (DSC) [31], layer priors (LP) [2], DetailNet [22], De-Raining Convolutional Neural Network (DRCNN) [34] and DID-MDN [33]. We retrain DRCNN with our data sets following the settings in the paper. Also, the codes of other methods are generously provided by the authors. We run all these methods on *Rain12* data sets and *New10* data sets to get the quantitative results. The average results are listed in Table 5. Note that columns ‘PSNR1’ and ‘SSIM1’ list the PSNR and SSIM values of *Rain12* data sets, while ‘PSNR2’ and ‘SSIM2’ list that of *New10* data sets.

As shown in the table, the proposed ROISEN has the highest PSNR and SSIM values on both data sets, which indicates that ROISEN has the best rain removal performance among these methods. As for DID-MDN, the published testing code generates output with a size of $512 \times 512$, therefore, we have to resize the output back to the original size to get the PSNR and SSIM values. This kind of operation would cause the loss of some information, hence leads to low-image quality and low-PSNR and -SSIM values. When testing on *New10* data sets, the SSIM value of DetailNet is close to that of ROISEN, but the PSNR value of ROISEN is greater than that of DetailNet.

Figs. 4 and 5 show the rain removal results of the mentioned deep learning-based methods on two images in *Rain12* data sets and *New10* data sets. As shown in Fig. 4*d*, the result of DID-MDN still contains rain streaks at a noticeable level, while DetailNet, DRCNN, and the proposed ROISEN remove most of them. Moreover, as shown in Fig. 4*f*, the result of ROISEN contains fewer rain streaks and reserves more details such as the whiskers of the animal highlighted with a red rectangle. Similar situations can be found in Fig. 5. In Fig. 5*d*, DID-MDN can remove some rain streaks while still leaving a lot of them in the result. The results of DRCNN are better than that of DID-MDN visually as shown in Fig. 5*e*, but still not good enough. Among these methods, DetailNet and ROISEN seem to have the best rain removal results, which are shown in Figs. 5*c* and *f*. As a reference, the PSNR and SSIM values of Fig. 5*c* are 34.16 and 0.9859, respectively, while that of Fig. 5*f* are 34.40 and 0.9871.

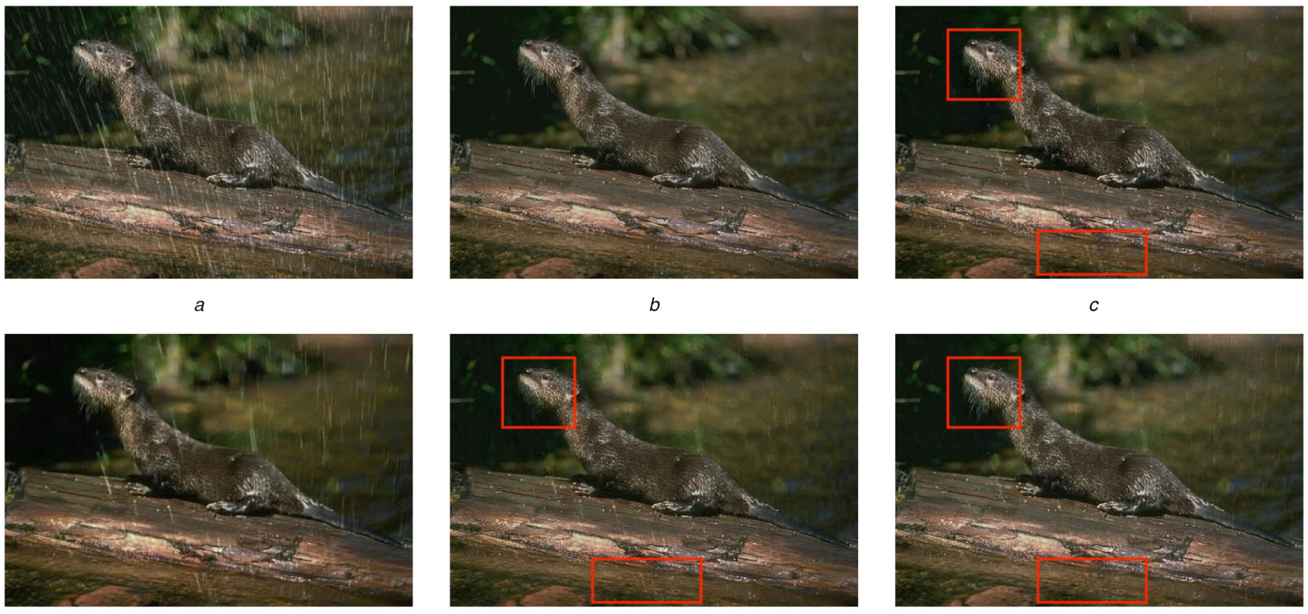

**Fig. 4** *Rain removal result of different methods on one image in Rain12 data sets*
***(a)*** Rainy, ***(b)*** Ground truth, ***(c)*** DetailNet [22], ***(d)*** DID-MDN [33], ***(e)*** DRCNN [34], ***(f)*** ROISEN

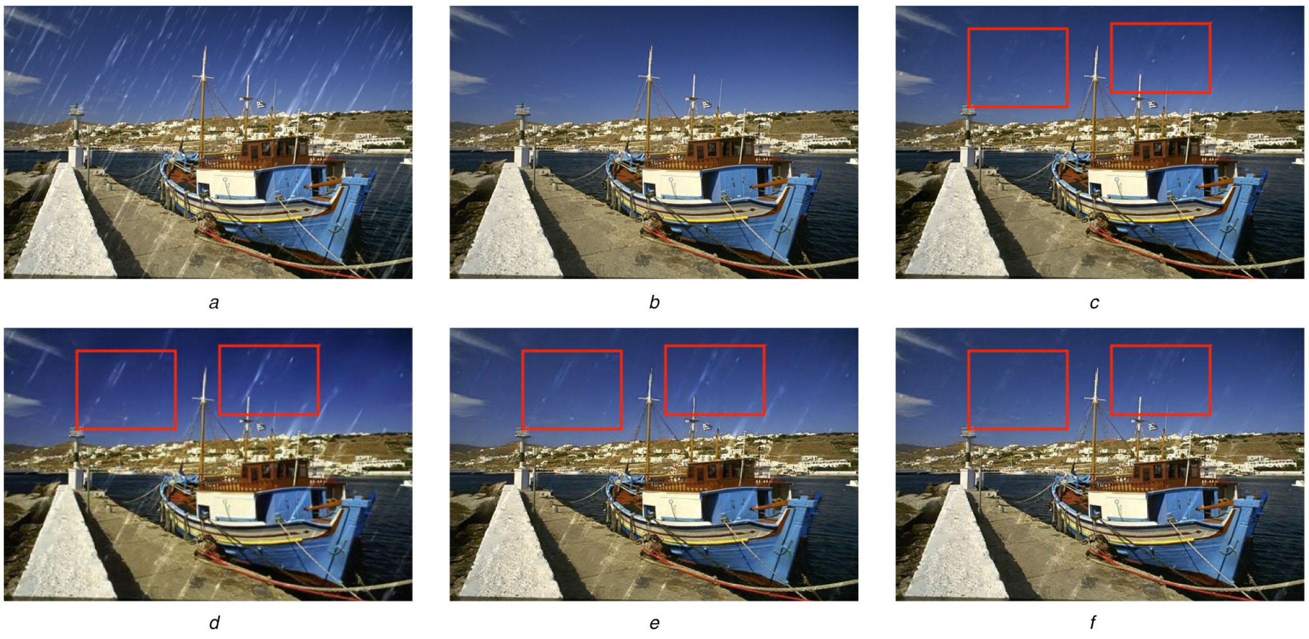

**Fig. 5** *Rain removal result of different methods on one image in New10 data sets*
***(a)*** Rainy, ***(b)*** Ground truth, ***(c)*** DetailNet [22], ***(d)*** DID-MDN [33], ***(e)*** DRCNN [34], ***(f)*** ROISEN

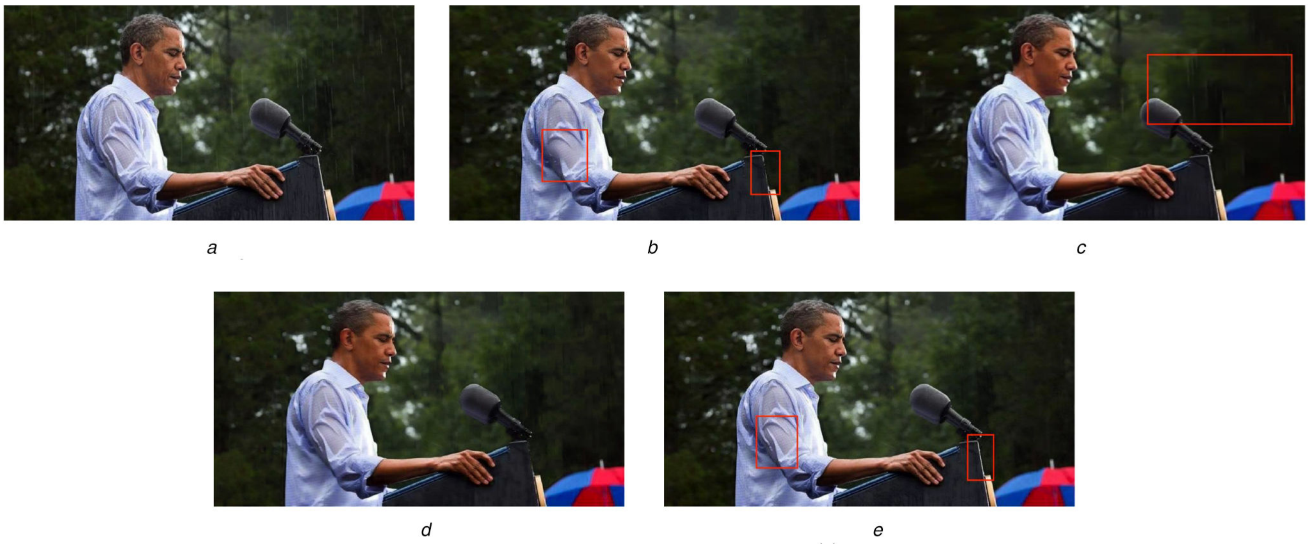

**Fig. 6** *Rain removal result of different methods on real-world image 'obama'*
***(a)*** Rainy, ***(b)*** DetailNet [22], ***(c)*** DID-MDN [33], ***(d)*** DRCNN [34], ***(e)*** ROISEN

### *4.4 Results on real-world images*

When tested on real-world rainy images, they do not possess the corresponding rain-free images. Thus, neither PSNR nor SSIM can be used as a quantitative measurement. Also, since there are no reliable no-reference image quality assessments exist yet, we evaluate all deep learning-based methods by the visual quality of their rain removal result in this section. Figs. 6 and 7 show two comparisons of different methods on two real-world images 'obama' and 'roof'.

As can be seen in Fig. 6*d*, for DRCNN, even though it could remove rain streaks, it introduces the blurry look to the result. As shown in Fig. 6*b*, the result of DetailNet contains fewer rain streaks. However, some details are lost as well, such as the wrinkles of the sleeve, and the edge of the rostrum beneath the microphone, which are highlighted with red rectangles in the figure. On the other hand, the proposed ROISEN can reserve most of the details and remove most of the rain streaks. Also, it does not cause over-smoothness or the blurry look in the result, as shown in Fig. 6*e*.

Similar situations can be found in Fig. 7. As shown in Figs. 7*c* and *d*, the result of DID-MDN contains lots of rain streaks, while DRCNN introduces the blurry look. The results of DetailNet and ROISEN suffer from over-smoothness slightly this time as well. One possible reason is that the rain streaks in the region under the roof are too dense and overlap each other, leaving limited background information available for reconstruction. Overall, the results of the proposed ROISEN have visual quality comparable to that of DetailNet, while better than that of most state-of-the-art methods including DRCNN and DID-MDN.

### *4.5 Running time*

Besides quantitative and qualitative comparison, we also make a comparison among the running times of all deep learning methods mentioned above. Note that in this section, the running time of deep learning methods means the testing time when running on GPU. We measure the running time of all four deep learning methods on different image sizes. Every method runs five times on each image size and the average running time is recorded in Table 6. All methods are run on a PC with MATLAB R2014a, six CPUs of Intel i7-8700, 32 GB memory and a GPU of NVIDIA GeForce 1080.

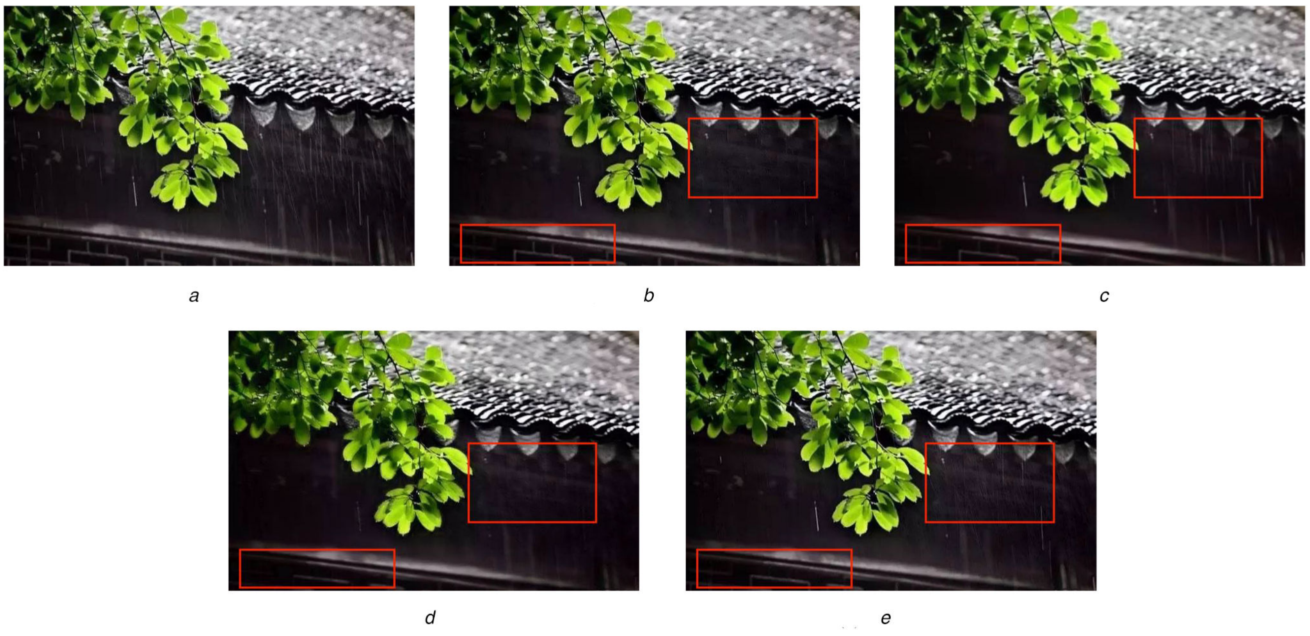

**Fig. 7** *Rain removal result of different methods on real-world image 'roof'*
***(a)*** Rainy, ***(b)*** DetailNet [22], ***(c)*** DID-MDN [33], ***(d)*** DRCNN [34], ***(e)*** ROISEN

**Table 6** Running time comparison of deep learning methods

| Methods | $300 \times 300$ | $500 \times 500$ | $800 \times 800$ |
|---|---|---|---|
| DetailNet | 0.16 s | 0.19 s | 0.24 s |
| DID-MDN | 0.12 s | 0.13 s | 0.14 s |
| DRCNN | 0.01 s | 0.02 s | 0.03 s |
| ROISEN | 0.04 s | 0.09 s | 0.20 s |

As listed in the table, DRCNN has the shortest running time for all three image sizes. This is expected since DRCNN contains two networks that add up to four layers. Also, the numbers of filters are small. An interesting finding is that for DID-MDN, the running time is almost the same for all sizes. This is probably because according to the published code, DID-MDN resizes the input into a fixed size first and then performs rain removal. As the image size increases, the running time of ROISEN is comparable to that of DetailNet. It can be noticed that all methods take no longer than half a second. The running time would be longer as the image size increases. However, it would still be pretty fast for the deep learning method to perform rain removal on a single image, especially with the aid of GPU. For comparison, when dealing with an image of size $800 \times 800$, DSC takes about 5 min, and LP takes about 18 min. Both are way longer than that of the deep learning methods.

## 5 Conclusion

In this study, we propose a ROISEN to address the problem of removing rain streaks from single images. First, we introduce a new network connection called ROI connection. ROI connection can provide more texture details for the network, which is crucial for the restoration after rain removal. SE block and BN are adopted to further improve rain removal performance. We train the proposed ROISEN with two different kinds of inputs: the rainy images and the detail layers and conduct a comparison between them, confirming that training with detail layers would improve the performance of ROISEN. Experimental results show that the proposed network has a comparable performance on both synthetic images and real-world images to state-of-the-art methods. Future work involves enlarging the data sets and developing a more accurate rain model.

## 6 Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (61370102 and 61772209), the Science and Technology Plan Project of Guangdong Province (2014A050503057 and 2016A050502050), and the Guangdong Natural Science Funds for distinguished young scholar (2014A030306050).

## 7 References

[1] Gao, Y., Shan, X., Hu, Z., *et al.*: 'Extended compressed tracking via random projection based on MSERs and online LS-SVM learning', *Pattern Recognit.*, 2016, **59**, pp. 245–254
[2] Li, Y., Tan, R.T., Guo, X., *et al.*: 'Rain streak removal using layer priors'. Comput. Vis. Pattern Recognit., Las Vegas, NV, USA, 2016, pp. 2736–2744
[3] Alotaibi, A., Mahmood, A.: 'Deep face liveness detection based on nonlinear diffusion using convolution neural network', *Signal Image Video Process.*, 2017, **11**, (4), pp. 713–720
[4] He, K., Zhang, X., Ren, S., *et al.*: 'Deep residual learning for image recognition'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 770–778
[5] Jia, B., Feng, W., Zhu, M.: 'Obstacle detection in single images with deep neural networks', *Signal Image Video Process.*, 2016, **10**, (6), pp. 1033–1040
[6] Lee, B., Erdenee, E., Jin, S., *et al.*: 'Efficient object detection using convolutional neural network-based hierarchical feature modeling', *Signal Image Video Process.*, 2016, **10**, (8), pp. 1503–1510
[7] Revathi, A., Kumar, D.: 'An efficient system for anomaly detection using deep learning classifier', *Signal Image Video Process.*, 2017, **11**, (2), pp. 291–299
[8] Li, H.: 'Deep learning for image denoising', *Int. J. Signal Process. Image Process. Pattern Recognit.*, 2014, **7**, (3), pp. 171–180
[9] Liu, D., Wen, B., Liu, X., *et al.*: 'When image denoising meets high-level vision tasks: a deep learning approach', arXiv preprint arXiv:170604284, 2017
[10] Satya, V., Prasad, K.S., Prasad, T.J.: 'Deep learning approach for image denoising and image demosaicing', *Int. J. Comput. Appl.*, 2017, **168**, (9), pp. 18–26
[11] Xie, J., Xu, L., Chen, E: 'Image denoising and inpainting with deep neural networks'. Int. Conf. on Neural Information Processing Systems, Siem Reap, Cambodia, 2012, pp. 341–349
[12] Wang, K., Zhang, D., Li, Y., *et al.*: 'Cost-effective active learning for deep image classification', *IEEE Trans. Circuits Syst. Video Technol.*, 2016, **27**, (12), pp. 2591–2600
[13] Cai, B., Xu, X., Jia, K., *et al.*: 'Dehazenet: an end-to-end system for single image haze removal', *IEEE Trans. Image Process.*, 2016, **25**, (11), pp. 5187–5198
[14] Gonalves, L.T., Gaya, J.O., Drews, P., *et al.*: 'DeepDive: an end-to-end dehazing method using deep learning'. Conf. on Graphics, Patterns and Images (SIBGRAPI), São Domingos, Brazil, 2017, pp. 436–441
[15] Ling, Z., Fan, G., Wang, Y., *et al.*: 'Learning deep transmission network for single image dehazing'. IEEE Int. Conf. on Image Processing, Phoenix, AZ, USA, 2016, pp. 2296–2300
[16] Ren, W., Liu, S., Zhang, H., *et al.*: '*Single image dehazing via multi-scale convolutional neural networks*' (Springer International Publishing, The Netherlands, 2016)

[17] Dong, C., Chen, C.L., He, K., *et al.*: ‘*Learning a deep convolutional network for image super-resolution*’ (Springer International Publishing, Switzerland, 2014)

[18] Guo, T., Mousavi, H.S., Monga, V: ‘Deep learning based image super-resolution with coupled backpropagation’. 2016 IEEE Global Conf. on Signal and Information Processing, Washington, DC, USA, 2017, pp. 237–241

[19] Tseng, C.W., Su, H.R., Lai, S.H., *et al.*: ‘Depth image super-resolution via multi-frame registration and deep learning’. Signal and Information Processing Association Summit and Conf., Jeju, Republic of Korea, 2017, pp. 1–8

[20] Huang, G., Liu, Z., Van Der Maaten, L., *et al.*: ‘Densely connected convolutional networks’. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017, pp. 4700–4708

[21] Fu, X., Huang, J., Ding, X., *et al.*: ‘Clearing the skies: a deep network architecture for single-image rain removal’, *IEEE Trans. Image Process.*, 2017, **26**, (6), pp. 2944–2956

[22] Fu, X., Huang, J., Zeng, D., *et al.*: ‘Removing rain from single images via a deep detail network’. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017, pp. 3855–3863

[23] Xia, H., Zhuge, R., Li, H., *et al.*: ‘Single image rain removal via a simplified residual dense network’, *IEEE Access*, 2018, **6**, pp. 66522–66535

[24] Garg, K., Nayar, S.K.: ‘Detection and removal of rain from videos’. Proc. 2004 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Washington, DC, USA, 2004, vol. 1, pp. I-528–I-535

[25] Bossu, J., Hautière, N., Tarel, J.P.: ‘Rain or snow detection in image sequences through use of a histogram of orientation of streaks’, *Int. J. Comput. Vis.*, 2011, **93**, (3), pp. 348–367

[26] Santhaseelan, V., Asari, V.K.: ‘Utilizing local phase information to remove rain from video’, *Int. J. Comput. Vis.*, 2015, **112**, (1), pp. 71–89

[27] You, S., Tan, R.T., Kawakami, R., *et al.*: ‘Adherent raindrop modeling, detection and removal in video’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016, **38**, (9), pp. 1721–1733

[28] Ren, W., Tian, J., Han, Z., *et al.*: ‘Video desnowing and deraining based on matrix decomposition’. IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017, pp. 2838–2847

[29] Liu, J., Yang, W., Yang, S., *et al.*: ‘Erase or fill? Deep joint recurrent rain removal and reconstruction in videos’. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 3233–3242

[30] Kim, J.H., Lee, C., Sim, J.Y., *et al.*: ‘Single-image deraining using an adaptive nonlocal means filter’. IEEE Int. Conf. on Image Processing, Melbourne, Australia 2013, pp. 914–917

[31] Luo, Y., Xu, Y., Ji, H.: ‘Removing rain from a single image via discriminative sparse coding’. IEEE Int. Conf. on Computer Vision, Santiago, Chile, 2015, pp. 3397–3405

[32] Deng, L.J., Huang, T.Z., Zhao, X.L., *et al.*: ‘A directional global sparse model for single image rain removal’, *Appl. Math. Model.*, 2018, **59**, pp. 662–679

[33] Zhang, H., Patel, V.M.: ‘Density-aware single image de-raining using a multi-stream dense network’. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 695–704

[34] Wang, M., Mai, J., Cai, R., *et al.*: ‘Single image deraining using deep convolutional networks’, *Multimedia Tools Appl.*, 2018, **77**, pp. 1–14

[35] Hu, J., Shen, L., Sun, G.: ‘Squeeze-and-excitation networks’. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 7132–7141

[36] He, K., Sun, J., Tang, X.: ‘Guided image filtering’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013, **35**, (6), pp. 1397–1409

[37] Jia, Y., Shelhamer, E., Donahue, J., *et al.*: ‘Caffe: convolutional architecture for fast feature embedding’. ACM Int. Conf. on Multimedia, Orlando, FL, USA, 2014, pp. 675–678

[38] Huynh Thu, Q., Ghanbari, M.: ‘Scope of validity of PSNR in image/video quality assessment’, *Electron. Lett.*, 2008, **44**, (13), pp. 800–801

[39] Wang, Z., Bovik, A.C., Sheikh, H.R., *et al.*: ‘Image quality assessment: from error visibility to structural similarity’, *IEEE Trans. Image Process.*, 2004, **13**, (4), pp. 600–612

[40] Yang, W., Tan, R.T., Feng, J., *et al.*: ‘Deep joint rain detection and removal from a single image’. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017, pp. 1357–1366

[41] Yang, W., Tan, R.T., Feng, J., *et al.*: ‘Joint rain detection and removal from a single image with contextualized deep networks’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019, **27**, p. 1